I C A S E R E P O R T

I/O BUFFER PERFORMANCE IN A VIRTUAL MEMORY SYSTEM

Stephen W. Sherman

Richard S. Brice

Report Number 76-2

February 2, 1976

INSTITUTE FOR COMPUTER APPLICATIONS

IN SCIENCE AND ENGINEERING

Operated by the

UNIVERSITIES SPACE RESEARCH ASSOCIATION

at

NASA'S LANGLEY RESEARCH CENTER

Hampton, Virginia

I/O BUFFER PERFORMANCE IN A VIRTUAL MEMORY SYSTEM

Stephen W. Sherman*
Institute for Computer Applications in Science and Engineering

Richard S. Brice
Department of Civil, Mechanical and Environmental Engineering
George Washington University

ABSTRACT

In this study we construct a simulator of a data base management system running in a virtual memory environment. We use the simulator to investigate the value of using an I/O buffer in this environment. The simulator is driven by trace data obtained with a software probe. The simulator is validated and is used to verify a theoretical model which predicts paging and disk access rates produced by use of an I/O buffer in a virtual memory environment. Results from a multi-factor set of simulation experiments are analyzed. The factors include three page replacement algorithms, four buffer management algorithms, five virtual buffer sizes, three values for real memory and six well known and widely differing distributions for creating sequences of requests to the simulated data base management system.

Acknowledgement

We wish to thank Professor Jim Browne for suggesting this area of research and Professors Browne and Arden for their comments on our initial research.

* On leave from the University of Houston

This paper was prepared as a result of work performed under NASA Contract No. NAS1-14101 while the first author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665. The work of the second author was supported by NASA Grant NGR-09-010-078.

I. INTRODUCTION

Computer programs that are I/O bound, such as data base management systems, often use part of primary memory as a storage area or buffer for data from secondary memory. If we assume the overhead to manage the data in primary memory is negligible and the buffer consumes otherwise unrequired primary memory, then the use of buffers can only improve performance due to the faster access to primary than to secondary memory. In a virtual memory system, the user is typically unable to control the assignment of his user space between primary and secondary memory. The use of a buffer in a virtual memory system may cause a decrease in performance due to competition for primary memory between the program and the buffer. Performance can also be degraded by a phenomenon known as double paging. The dynamics of double paging was characterized by Goldberg and Hassinger [1] as the running of a paged operating system under a paged virtual machine monitor. We define double paging as the management of buffer storage under the control of a paged virtual memory environment.

We refer to the I/O buffers in a program running on a virtual memory system as virtual buffers. In a previous examination [2] of the use of virtual buffers, we conducted a series of statistically designed and analyzed experiments to study the effects of four

factors on performance. The factors were virtual buffer replacement algorithm, virtual buffer size, primary memory size and paging replacement algorithm. This series of 240 experiments was conducted by running a data base management program on a dedicated system and measuring the performance as we varied the factors. The data base management program executed a predetermined and unvarying script. The environment in which these experiments were conducted is similar to the controlled laboratory environment described in [3]. We will refer to these experiments as the test bed experiments.

The distribution of I/O requests is a function of the data base content and structure, the data base management system and the script. It is impossible to vary the distribution of data base requests in a controlled manner in the test bed experiments. In this study we vary the distribution of data base requests in a controlled environment through simulation to extend our previous results and investigate the effects of the distribution of data base requests on the performance metrics used in our previous study. Simulation is the only technique available for extending our previous investigations to include controlled distributions of requests.

Our simulator uses trace data as defined in [4] from a modified test bed experiment to simulate the paging effects of the operating system and data base management system. The trace data is also used to determine when to request information from the data base. The location of the information in the data base can be obtained from the trace data or can be generated in a controlled manner. The simulator is validated by comparing its performance to the test bed experiments. The simulator is also used to

verify a theoretical model developed in the previous study. A set of simulation experiments are then run which repeat the test bed experiments for six different data base request distributions.

We compare various performance metrics from our test bed experiments to our simulation experiments. Many observed trends in the performance data of the test bed experiments occur again in the performance analysis of our simulation experiments for various distributions. We relate the differences in the performance data to the characteristics of the distributions. The simulation experiments show that our test bed results are not unique due to peculiarities in the original data base request distribution. The differences in the performance of the simulation experiments and the test bed results are comprehensible and reasonable and generate confidence that the simulator is a good tool to investigate further extensions of our test bed experiments.

II. ENVIRONMENT AND THE COLLECTION OF TRACE DATA

We collected the trace data for the simulation program by instrumenting an operating system with a software probe. The application program, a prototype data base management (DBM) system, was run on a dedicated machine with the software probe collecting significant events on tape for later analysis. The DBM system executed a script of data base requests. The script completely traversed the data base and caused reading, insertion and deletion of data as it was executed.

The DBM system organizes data in a tree structured format. Requests are made in a series of interactive primitive functions that perform elementary operations upon the data base. The data, names and pointers are encoded in 40 word segments. These 40 word segments can be target segments or links to target segments and will be referred to as data base segments. One physical record on disk contains eleven data base segments (440 words in 1 record) and information is read from the disk in physical records. Even though our record size is smaller than the page size (512 words) we store each record on a different virtual buffer page to avoid physical page boundary overlap. The DBM virtual buffer size is fixed when the DBM program is initiated and can consume up to 32K words of virtual memory. The data base accessed by the script is a 7 level tree structure and consists primarily of integer and floating point numbers describing the geometry of an aircraft structure.

The DBM system was executed on a multi-user disk operating system running on a PRIME 300 minicomputer. The PRIME 300 can support up to 256K words (16 bits per word) of memory when utilizing its virtual memory capability. Our PRIME 300 has 64K words of memory and uses 2 moving head disks for paging and file storage. Each user is given a virtual address space of 64K words. A more detailed description of the environment can be found in [2].

The software probe is located in a page that the system has locked in memory and can not interact with the paging process. The probe records events that cause a significant change in the system. Examples of these events are: 1) a user issues

an SVC (service request); 2) a request is made to a peripheral device; and 3) a reference bit has been set. Every virtual page has a reference bit associated with it. A reference to a page whose reference bit is not set causes the reference bit to be set. All reference bits are reset whenever a page fault occurs.

Our trace data consists entirely of the reference bit events. The data collected by the probe at the same time the reference bit was set contains information sufficient to deduce: 1) whether a page fault occurred, 2) the virtual page and real page being referenced, 3) the owner of the virtual page, 4) the time and 5) the virtual page and real page being replaced if a page fault occurred. The origin of the DBM virtual buffer was fixed at a particular location so that references to the virtual buffer could be identified in the trace data. The size of the virtual buffer is chosen large enough to contain the entire data base on the run that gathered the trace data. Although gathering the data needed for our simulator does not require that the entire data base fit in the virtual buffer, it is more convenient to collect the data in this configuration. The choice of the virtual buffer manager does not affect the trace data since no records are replaced in the virtual buffer.

The run that gathered the trace data was configured to use the least amount of real memory possible (32K) so that the page faults and therefore the number of reference bit events would be as high as possible. We used the standard first in - first out (FIFO) page replacement algorithm in gathering the trace data.

A total of 52,341 reference bit events were generated of which 6716 were accompanied by page faults. Although most strings of reference events between page faults were quite short, there was an occasional long string of reference events between page faults.

The FIFO page replacement algorithm was modified to clear the reference bits of all the virtual pages if 10 consecutive reference events occurred without a page fault. This eliminates the long strings of reference bit events. The trace data which was used in all the simulations and validation experiments was collected with the modified FIFO page replacement algorithm. A total of 372,287 reference events were generated of which 6709 were accompanied by page faults. Instrumentation of the DBM showed that there were 13,996 references to data base segments. These references would cause 1075 disk accesses if only a one page virtual buffer were available since the data contained 1075 record transitions. The trace data had 13,831 references to data base segments distributed over a sequence of 1071 record transitions.

III. DESCRIPTION OF THE SIMULATION MODEL

The simulation model consists of an initializer, virtual buffer manager, page replacement algorithm and performance reporter. The program is initialized with a real memory size and virtual buffer size. Various paging and buffer tables are then initialized. The trace data consists of the reference string. Each element of the string is read by the model and treated as a virtual page address. If the virtual page address is not a virtual buffer reference, it is passed to the page replacement algorithm.

If the virtual page address is a virtual buffer reference, the buffer manager must insure that the corresponding record is in the virtual buffer. If the record is in the virtual buffer, the virtual page address of the virtual buffer in which the record is located is passed to the page replacement algorithm. If the record is not in the virtual buffer, the buffer manager selects a record to be replaced and puts the new record in its place. The virtual page address of the virtual buffer in which the new record is placed is passed to the page replacement algorithm.

The page replacement algorithm compares the virtual page address with the virtual pages in real memory. If the virtual page is found in real memory, the virtual page reference is noted and the next virtual page address is requested from the trace data. If the referenced virtual page is not found in real memory, the page replacement algorithm removes a virtual page from real memory and replaces it with the referenced virtual page. The next virtual page address is requested from the trace data. The flow is summarized in Figure I.

When the trace data is exhausted the performance reporter presents the statistics gathered during the run. These statistics include; number of page faults, average number of real pages used by the virtual buffer (average buffer size), number of times a reference to a record in the virtual buffer caused a page fault (reference faults), the number of times the virtual buffer manager caused a page fault when it tried to replace a record (double page fault), and the number of times the virtual buffer manager read in a record (I/O access).

IV. VALIDATION OF THE MODEL

The simulation model described in the previous section is extremely simple. The trace data does not contain all the virtual page references although similar data has been used successfully to simulate paging in the CP-67 simulator [5]. To validate the simulator, we simulate the 240 test bed experiments that we had previously run. We use 3 page replacement algorithms; first in-first out (FIFO), random (RAND) and second chance [6] (SCH). The SCH algorithm is also known as the Multics algorithm [7] and the use bit algorithm [8]. We simulate 4 virtual buffer managers; FIFO, RAND, SCH and least recently used (LRU). We choose virtual buffer sizes of 1, 5, 10, 15 and 20 pages. We set the total amount of real memory available to the system to 36K, 40K, 44K and 48K.

The four factors, virtual buffer size, real memory size, virtual buffer manager and page replacement algorithm have five, four, four and three levels of interest respectively. We simulated the complete 5 by 4 by 4 by 3 factorial experiment consisting of all 240 combinations.

To compare the simulated factorial experiments and the test bed factorial experiments we use 5 primary measures and 5 secondary measures of system performance. The 5 primary measures are numbers of page faults, reference faults, double page faults, I/O accesses and average buffer size. The 5 secondary measures are; number of times the reference bit is set without causing a page fault (reference sets), the average number of page faults

outside the virtual buffer between page faults within the virtual buffer (mean), the standard deviation for the number of page faults outside the virtual buffer between page faults within the virtual buffer (standard deviation), the number of times a page in the virtual buffer replaces a page not in the buffer (BRP) and the number of times a page in the virtual buffer is replaced by a page not in the buffer (PRB).

With the exception of I/O accesses, relative errors between corresponding measures are computed for all 240 experiments. We define the relative error to be 100* ABS [(SIMULATED MEASURE - TEST BED MEASURE)/ABS (TEST BED MEASURE)].

The average values of the relative errors for all of the primary measures except I/O accesses are shown in Figure II. The average values of the relative errors for the secondary measures are displayed in Figure III. A comparison of I/O accesses is presented separately in Table I since the I/O accesses are invariant with respect to the real memory size and paging algorithm. A sample set of data from the test bed and the simulation is presented in Table II.

Some of the average relative errors for the smaller virtual buffer sizes are rather large. An examination of Table II shows that the value of a particular measure may range over several orders of magnitude. The relative errors of small numbers tend to be large in our comparisons due primarily to differences in initial conditions of the test bed and simulation experiments. For example differences in reference faults in Table II cause relative percentage errors of 75%, 31%, 18%, 6%, and 3% for virtual buffer

sizes 1 through 20. The change in magnitudes of the values for most of the measures presented in Figures II and III account for their high average relative error in the first few buffer sizes. The average relative errors at virtual buffer sizes 10, 15, and 20 give a more accurate account of the close correspondence between simulated experiments and the test bed experiments. The mean and standard deviation measures have high relative errors in the cases with small virtual buffer sizes because they are calculated with only a few samples (number samples = reference faults + double page faults) when the virtual buffers are small.

The simulation of the test bed experiments yields performance results that are generally in good agreement with the test bed results. All of the known trends in the test bed performance are observed in the performance data of the simulation experiments. The close correspondence of the simulation results and the test bed results for all 240 different experiments demonstrates the validity of the simulator.

V. VERIFICATION OF THE THEORETICAL MODEL

In our previous paper [2], we developed a very simple theoretical model which predicts total I/O per data base request (T) in the virtual buffer as a function of virtual buffer size in pages (N), pages of real storage available for the virtual buffer (M) and number of pages in the data base (D). The model assumes that random data base requests are uniformly distributed, uses the RAND page replacement algorithm and the RAND buffer manager. Total I/O in the virtual buffer is given by

$$T(M,N,D) = (1 - \frac{M}{N}) \text{ page faults} + (1 - \frac{N}{D}) \text{ I/O accesses}$$

for $1 \leq M \leq N \leq D$.

The test bed experiments were unable to verify the accuracy of the model since we could not control the distribution of data base requests. However, we did verify that the trends predicted by the model were followed in the test bed experiments.

With the simulator, we are able to investigate the combination of RAND page replacement, RAND buffer manager and a uniform distribution of random requests for data base segments. Competition for real memory from the program and system and the amount of real memory available are not explicitly included in the theoretical model although they are significant factors in the simulator and the trace data. The simulator uses a global paging algorithm and does not allocate a fixed amount of real memory to the virtual buffer. The performance measure, average buffer size, computed by the simulator is used as M in the model. We contend that using average buffer size in place of M in the theoretical model should reflect the value for the amount of real memory in the virtual buffer and compensate for the competition for the real memory between the buffer and the program.

Table III contains the results of the performance of the simulator and the predictions of the model using the average buffer sizes calculated by the simulator for M. The close agreement in the table values indicates that the observed average buffer size does reflect the value for the amount of real memory in the virtual buffer and competition for real memory between the program and the buffer.

VI. SIMULATION EXPERIMENTS

The simulation experiments are very similar to the validation experiments. The same 3 page replacement algorithms, 4 buffer management algorithms and 5 virtual buffer sizes are simulated. The three largest real memory sizes are simulated. We decided not to simulate the 36K real memory size because our test bed experiments indicated the performance trends were very similar to the experiments with the 40K memory size.

For these experiments, the simulator uses the same trace data used in the validation experiments. Data base access requests are replaced by requests generated from one of six distributions. The data base is still treated as though it was 45 pages long and consiste of 495 data base segments. Three of the six data base request distributions generate string oriented requests. The other distributions are well known and we picked these distributions because they differed significantly from our original trace data distribitions More realistic models for data base reference strings [9] can easily be adapted as input for our simulator.

Three of our distributions are the random (RA1), binomial (BI1) and poisson (PO1). A sample from 1 to 495 is generated from one of these three distributions whenever the trace data indicates that an access to the data base is required. The sample is translated into a disk record from 1 to 45 and the buffer manager is presented with the request. The string oriented distributions are generated by treating the 495 data base segments as if they were in a binary tree with the root node being data segment 1, the 2 nodes attached

to data segment 1 are nodes 2 and 3, and continuing down nine levels where the nodes are numbered 256 to 495. We generate a string of 9 data base segment references from this tree by generating a number from 256 to 495 by sampling from the random (RAS), binomial (BIS) or poisson (POS) distributions. The sample and the 8 nodes that must be traversed to reach that sample constitute our 9 string sequence of data segments. The string of 9 data segments maps into 6 or 7 distinct disk records and they are issued in sequence to the buffer manager during this request for a data base access and the next 8 requests.

The five experimental factors, virtual buffer size, real memory size, buffer management algorithm, page replacement algorithm and distribution of data base requests have 5, 3, 4, 3 and 6 levels of interest respectively. There are 1080 possible combinations and all of the combinations are simulated.

VII. RESULTS OF SIMULATION EXPERIMENTS

It is impossible to present the mass of data generated by all of the simulation experiments. Representative figures and tables will be provided in order to omit congestion. Whenever possible our technique in presenting the simulation results will be to use the test bed results in [2] as a focus for comparison. In our analysis we distinguish between I/O in the virtual buffer and I/O resulting from program paging. The buffer I/O is further classified as I/O accesses, double page faults and reference faults. Total I/O in the virtual buffer will be discussed after examination of its components. Finally, the total cost of executing the script will be compared to the test bed results.

The I/O access for the test bed results are presented in Table I. The better performance of the RAND buffer manager at virtual buffer sizes 10 and 15 was an unexpected result. An analysis of the original data base reference string distribution indicates that strings of references of length 10 to 15 often separated references to frequently referenced records. While the RAND buffer manager might allow the frequently referenced records to remain in the buffer, the other buffer managers were forced to replace them with the sequence.

The simulation experiments support our explanation. In the experiments with the RA1, RO1, and BI1 distributions, the RAND buffer manager consistently requires more I/O accesses than the other buffer managers.

In the simulation experiments using the string oriented distribution, the RAND buffer manager needs fewer I/O accesses than any of the other buffer managers when the virtual buffer size is 5. At virtual buffer sizes 10, 15, and 20 the RAND buffer manager requires a relatively large number of I/O accesses compared to the others. In our string oriented distributions, the string always required 6 or 7 record transitions which is long enough to make the RAND buffer manger advantageous only at a virtual buffer size of 5. The I/O accesses for the PO1 and POS distributions are shown in Table IV.

Double page faults have previously been defined as the number of times the virtual buffer manger causes a page fault when it tries to replace a record. The double paging rate is the number of double page faults divided by the number of times the virtual buffer manager

requires an I/O access. The test bed experiments show that the RAND buffer manager consistently has the lowest double paging rate and that the lowest double paging rate occurs when the RAND buffer manager is combined with the RAND paging algorithm.

The simulation results again confirm the test bed results for the double paging rate. The double paging rates of the FIFO, SCH, and LRU algorithms are barely distinguishable in the test bed experiments. Figure III illustrates that different distributions can cause a variance in the double paging rate of those previously indistinguishable algorithms.

The double paging rate for the RA1 distribution is similar to the test bed results. The largest variance of double paging rate among buffer managers is caused by the POS distribution. The PO1 and RAS distribution are second and third largest in this respect. When the virtual buffers are large, the BI1 and BIS distribitions do not have enough I/O accesses to produce meaningful numerical comparison of their double paging rates with those of the other distributions.

Reference faults were defined as the faults caused by references to data contained in the virtual buffer.
The reference paging rate is defined as the number of reference faults divided by the number of record transitions minus the number of I/O requests to the disk. In the test bed experiments, the RAND buffer manager has a higher reference paging rate than the other buffer managers. As the real memory size increases the difference disappears. The buffer managers have similar properties when the RAS and RA1 distributions are used. Figure V illustrates the higher double paging rate for the RAND buffer manager when the RA1

distribution is used. The double paging rate of the RAND buffer manager remains noticeably higher than the rest when the comparisons are made at higher memory sizes. The buffer managers using the PO1 and POS algorithms have approximately the same reference paging rate as illustrated in Figure VI for the PO1 distribution using 44K of memory. The similarity in reference paging rate does not change as the real memory size is increased using the PO1 and POS distributions.

The number of page faults in the virtual buffer (reference faults + double paging faults) is only slightly affected by the buffer manager in the test bed experiments with the RAND buffer manager incurring fewer page faults in the buffer than the others. The range of values for page faults in the virtual buffer are similar and the buffer managers behave similarly for the test bed experiments and the simulation with the BIS distribution. The simulations of the other distributions almost doubles the range of values for the number of page faults in the buffer. The RAND buffer manager often incurs fewer buffer page faults than the other buffer managers. The percent difference in buffer faults between the RAND buffer manager and the other buffer managers is an increasing function of the total number of buffer faults for a given memory size.

Total I/O in the virtual buffer consists of three components: reference faults, double paging faults and I/O accesses. In our analysis of total I/O we do not distinguish between disk accesses and page faults. The theoretical model predicts that total I/O can decrease as the virtual buffer size is increased. For all test bed experiments and for all simulation experiments,

total I/O in the buffer reaches a minimum when the virtual buffer size is 20 pages. The total I/O in the buffer for the RA1, B11, and PO1 distributions decreases monotonically with increasing virtual buffer size as illustrated in Figure VII. In all the test bed experiments and for the simulation experiments with string oriented distributions, the total buffer I/O for all buffer managers except RAND is an increasing function for small virtual buffer sizes and a decreasing function when the buffers are large. The monotonically decreasing behavior of the RAND buffer manager is shown in Figure VII using the RAS distribution.

The average buffer size, real memory in the virtual buffer, increases as the virtual buffer size increases in the test bed and simulation experiments. This increase causes more page faults in the program and the system. The increase in program page faults must be combined with any decrease in total I/O in the buffer to produce a total cost of executing the script (execution cost). The execution costs closely parallel the results on total I/O. Those distributions whose total I/O decreases monotonically have execution costs that are similar but do not decrease as quickly. The string oriented distributions and test bed experiments show increases in execution cost corresponding to the increasing total I/O values at small virtual buffer sizes.

In all the simulation experiments the execution cost is lower at the largest virtual buffer size than when only one virtual buffer is used. The test bed experiments usually have

a slightly higher execution cost when using the largest virtual buffer size.

The distribution of data base requests in the test bed experiments have longer sequences of strings and cause almost an order of magnitude fewer record transitions than the simulation distributions. The longer strings cause total I/O to continue to increase for a larger range of virtual buffer size in the test bed experiment. The lack of record transitions prohibits large decreases in the cost of execution from the reduction of I/O accesses.

In an effort to analyze the variation in results due to double paging, we apply analysis of variance to all the performance measures described in this section. For most of the performance measures, either the virtual buffer size or the main memory size cause a significant amount of the variation. Similar results concerning the influence of main memory are reported in [10]. If the double paging phenomenon were significant we would expect the interaction of the paging algorithm and buffer manager to be significant compared to their individual effects on the variance. We have not found the interaction to be a significant factor in the simulation experiments or the test bed experiments.

VIII. CONCLUSION

We constructed a simulation model to extend our previous results on the use of virtual buffers. The model used trace data gathered from a real system. Although a complete reference string was not used, we showed through our validation

that the partial reference string was sufficient to validate a comprehensive set of 240 experiments using 3 different paging algorithms.

The simulation model was also used with a uniform distribution of random data base requests to verify a theoretical model. The verification showed that the average buffer size performance metric calculated by the simulator compensated for the interference effects of program paging in the virtual buffer.

An extensive set of simulation experiments was conducted to compare their performance with the performance of a set of test bed experiments conducted in a laboratory environment. The test bed experiments all used the same data base manager and script.

Simulation was required in order to vary the distribution of data base requests in a controlled environment. The test bed experiments and the simulation results for string oriented distributions had a similar effect on the I/O accesses generated by the RAND buffer manager. The double paging rates, reference paging rates and number of page faults in the buffer were comparable for the simulation experiments and the test bed experiments. The total I/O in the virtual buffer is similar for the simulations using string oriented distributions and the test bed experiments. The execution costs of the simulation studies are less than the cost in the test bed experiment, but the differences are understandable when certain properties of the simulated distributions are considered. The analysis of variance showed no significant interaction of

the paging algorithm and buffer manager over a number of metrics in both the simulation results and test bed results.

The general agreement between the simulation results and the test bed results reinforces our previous studies. The differences that appeared were reasonable and justifiable. The simulator supported our previous identification of those performance factors which were dependent on the script. We gained confidence in the simulator as a tool for further experimentation in the use of virtual buffers.

BIBLIOGRAPHY

1. Goldberg, R. and R. Hassinger, "The Double Paging Anomaly," Proc. 1974 National Computer Conference, Chicago, May 6-8, 1974.

2. Sherman, S. W. and R. S. Brice, "Experiments Concerning Buffer Management in a Virtual Memory System," ICASE Report 75-20, Langley Research Center, Hampton, Va., Oct. 1975.

3. Schwetman, H. D. and J. C. Browne, "An Experimental Study of Computer System Performance," Proc. National ACM Conference, Boston, Mass., August 1972, pp. 693-703.

4. Sherman, S. W. and J. C. Browne, "Trace-Driven Modeling: Review and Overview," Symposium on the Simulation of Computer Systems, Gaithersburg, MD, June 1973, pp. 201-208.

5. Boksenbaum, C.;S. Greenberg, and C. Tillman, "Simulation of CP-67," IBM Report 320-2093, June, 1973.

6. Hoare, C. A. R., and R. M. McKeag, "A Survey of Store Management Techniques," A.P.I.C. Studies in Data Processing, No. 9, Academic Press, 1972.

7. Corbato F. J., "A Paging Experiment with the Multics System," In Honor of P. M. Morse, M.I.T. Press, Cambridge, Mass., 1969, pp. 217-228.

8. Grit, D. H. and R. Y. Kain, "An Analysis of a Use Bit Page Replacement Algorithm," Proceedings ACM Annual Conference, 1975.

9. Easton, M. C., "Model for Interactive Data Base Reference String," IBM Report RC 5050, Sept. 1974.

10. Tsao, R. F.; L. W. Comeau, and B. H. Margolin, "A Multi-Factor Paging Experiment I, II," Statistical Computer Performance Evaluation, edited by Walter Freiberger, Academic Press, pp. 103-158.

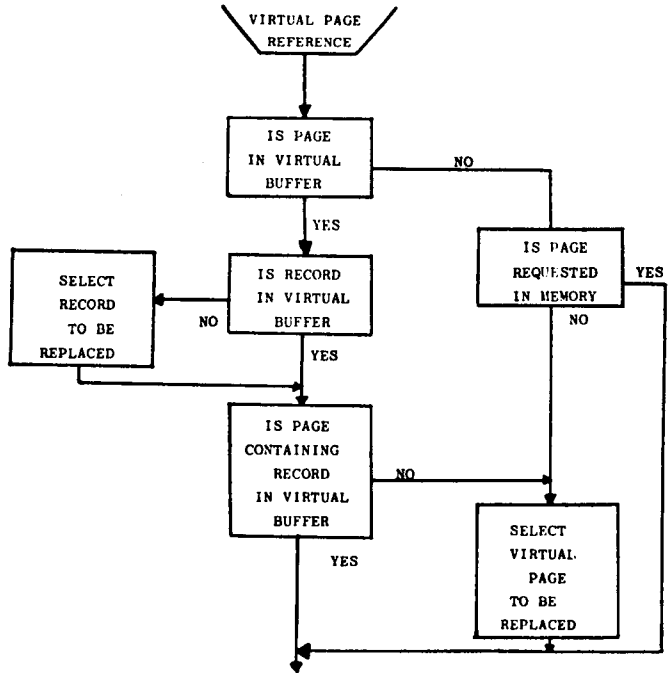

FIGURE I

PROCESSING OF TRACE DATA BY THE SIMULATOR

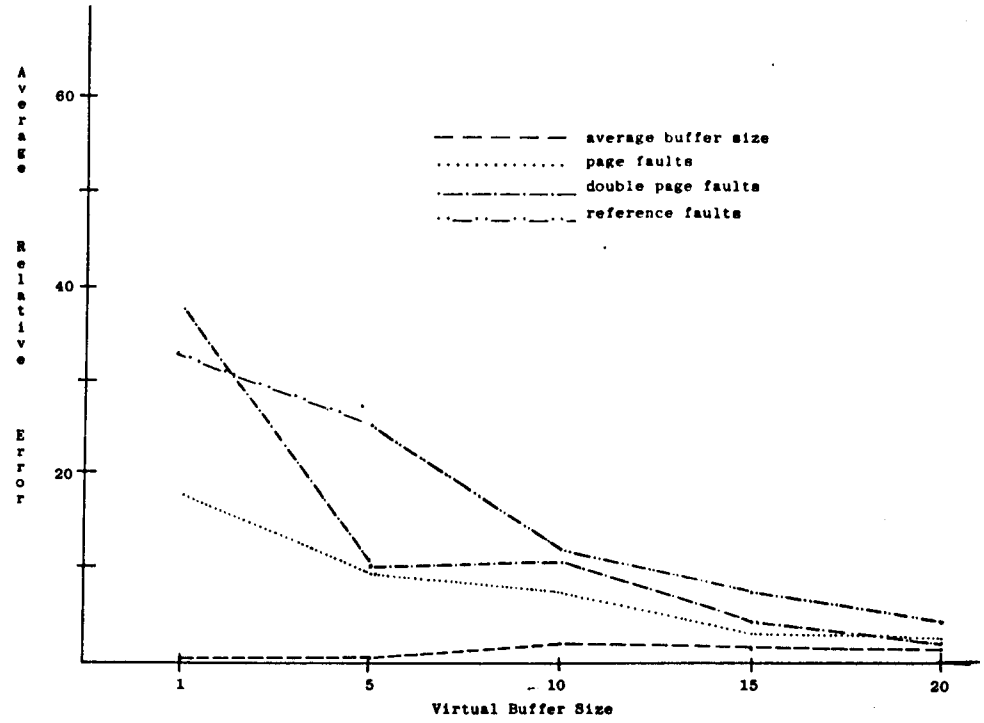

Figure II

Average relative error for the primary measures

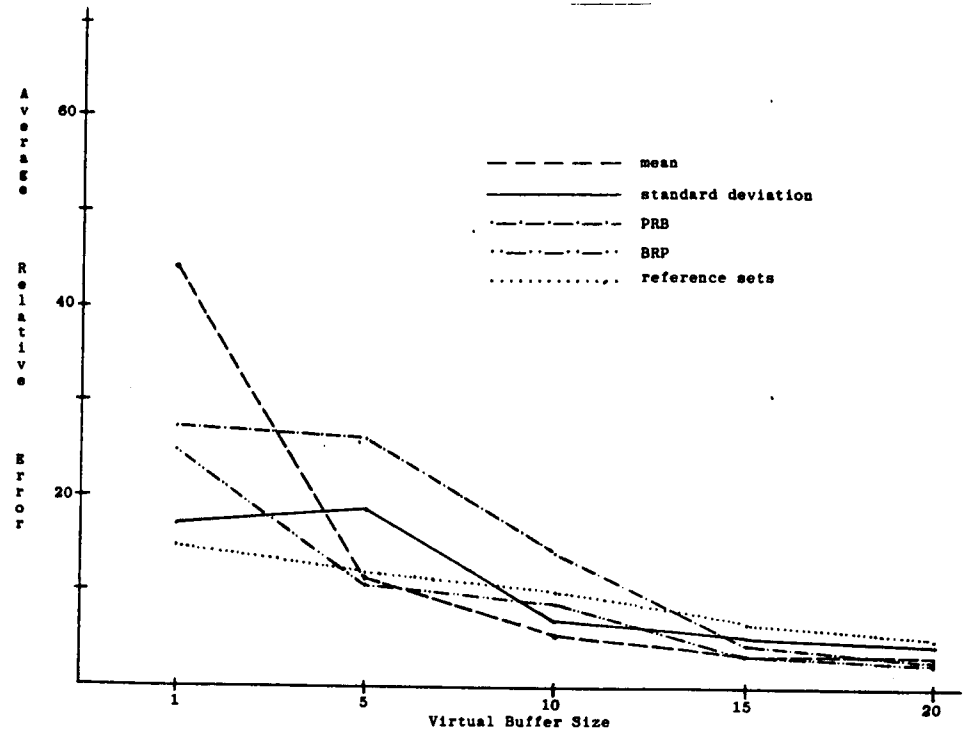

Figure III

Average relative error for the secondary measures

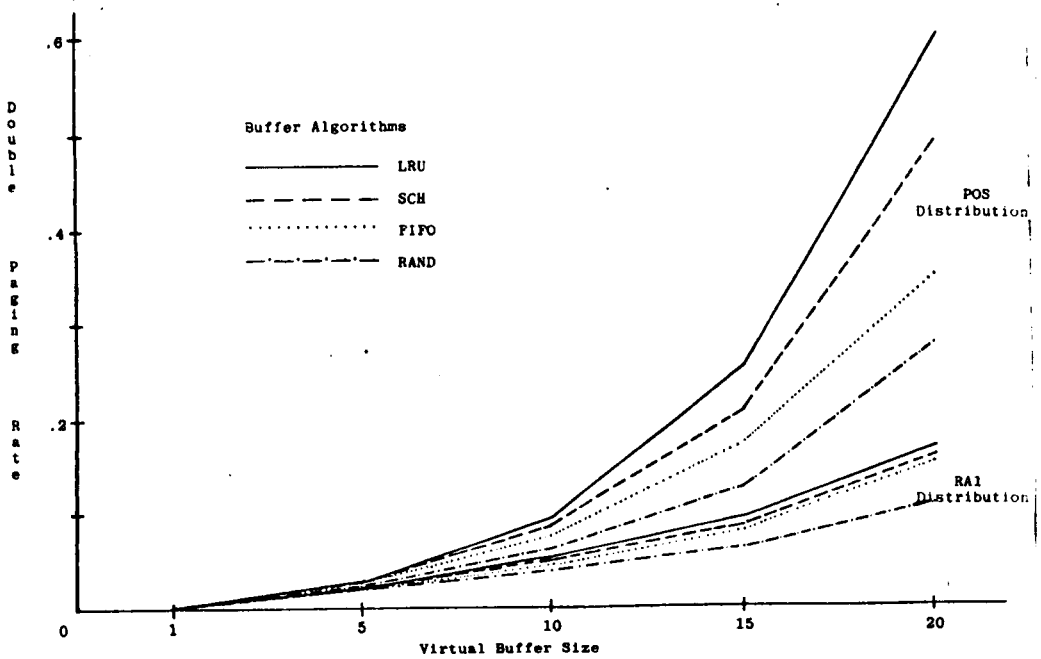

Figure IV

Double paging rate for the FIFO paging algorithm using 40K of real memory.

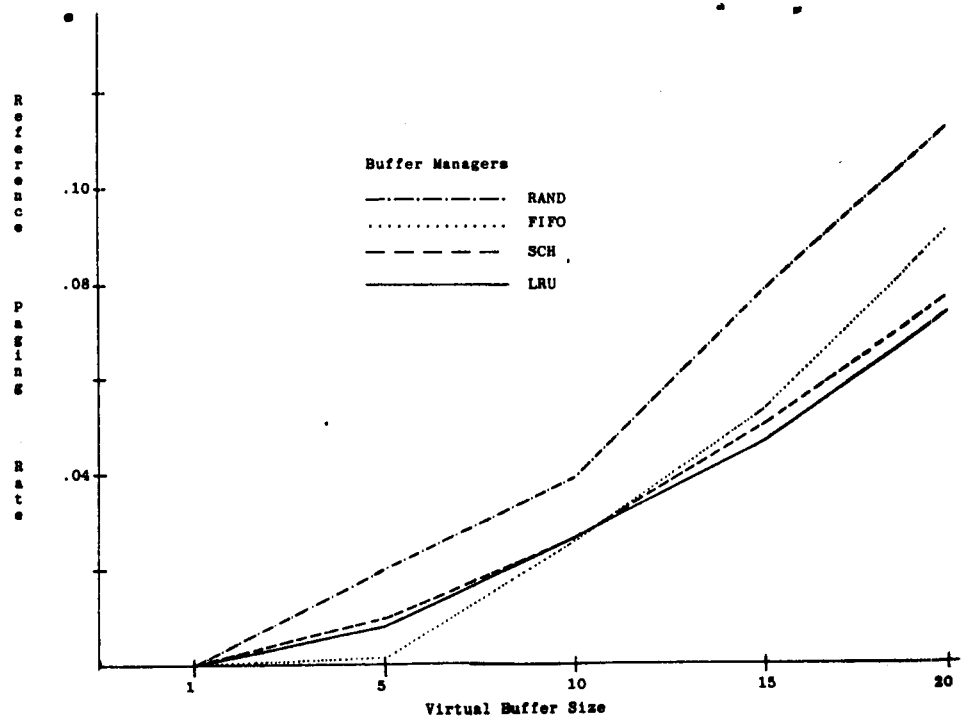

Figure V

Reference paging rate of the buffer managers using the FIFO paging algorithm and 44K of real memory with the RA1 distribution.

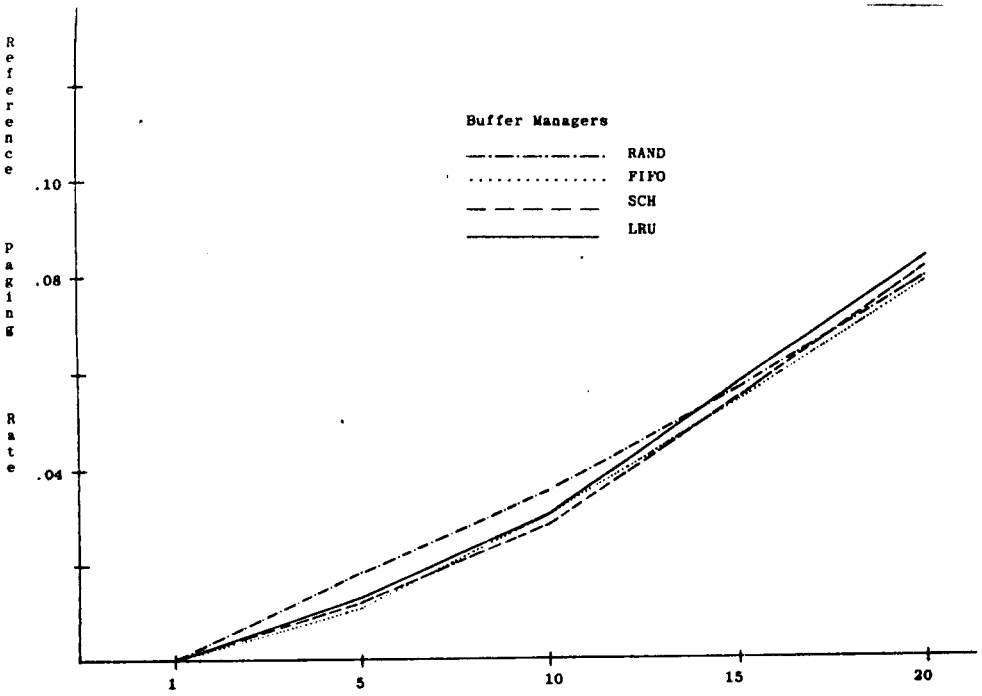

Figure VI

Reference paging rate of the buffer managers using the FIFO paging algorithm and 44K of real memory with the PO1 distribution.

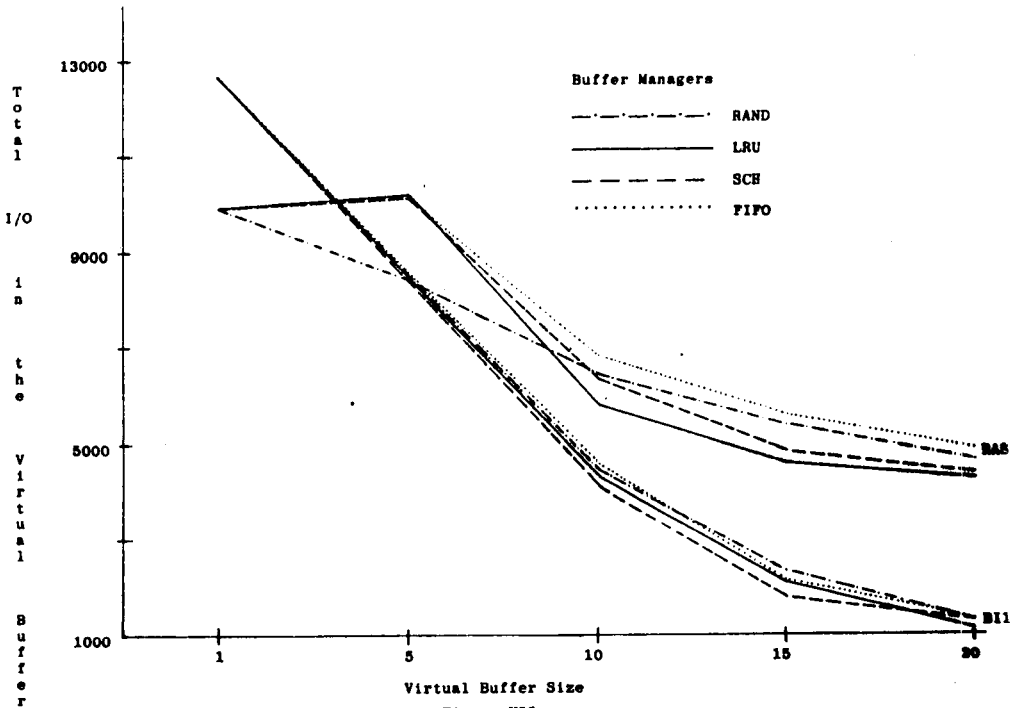

Figure VII

Total I/O in the buffer for the FIFO paging algorithm using 40K with the RAS and BI1 distributions.

VIRTUAL BUFFER SIZE	BUFFER MANAGERS: TEST BED LRU	TEST BED SCH	TEST BED RAND	TEST BED FIFO	SIMULATION LRU	SIMULATION SCH	SIMULATION RAND*	SIMULATION FIFO
1	1075	1075	1075	1075	1071	1071	1071	1071
5	784	791	774	794	787	794	779	797
10	684	687	559	688	687	696	534	690
15	418	422	270	434	421	425	283	437
20	93	98	154	103	96	101	165	106

TABLE I

Comparison of the number of I/O accesses generated by the buffer managers in the test bed and simulation experiments.

*RAND values are averages taken from 4 different seeds in the random number generator.

	VIRTUAL BUFFER SIZE 1	5	10	15	20
Page faults	263 237	662 576	1276 1262	1768 1798	1932 1900
Average buff size	1.00 .99	4.84 4.82	8.84 8.86	11.45 11.22	11.98 11.81
Reference faults	4 1	13 9	49 58	129 137	294 286
Double page faults	3 6	42 39	131 118	194 195	84 88
Reference sets	4705 4617	11632 10198	18052 17124	22464 21313	22824 22154
Mean	33.43 28.50	10.84 10.69	6.04 6.12	4.44 4.40	4.10 4.07
Standard deviation	27.82 38.02	10.27 12.04	7.06 7.43	6.42 7.06	6.85 6.83
BRP	7 7	50 43	158 156	270 266	318 310
PRB	7 6	45 38	150 148	261 257	308 301

TABLE II

Sample set of measures for validation with test bed data followed by simulated data for RAND paging algorithm, FIFO buffer manager and 44K of real memory.

I/O ACCESSES

Virtual Buffer Size	Simulator: Memory Sizes 36K	40K	44K	48K	Model (13831 data base requests with D=45)
1	13528	13532	13528	13529	13523
5	12249	12322	12292	12299	12294
10	10781	10817	10734	10730	10758
15	9276	9235	9152	9248	9222
20	7739	7790	7731	7798	7684

PAGE FAULTS IN THE VIRTUAL BUFFER

	Simulator 36K	40K	44K	48K	Model: Using the M corresponding to the memory size 36K	40K	44K	48K
1	59	27	6	3	0	0	0	0
5	395	205	58	15	358	193	55	27
10	930	542	253	55	824	538	248	41
15	1582	1043	546	211	1603	958	561	193
20	2507	1622	1048	476	2469	1611	1002	462

TABLE III

Simulator values for RAND paging algorithm, RAND buffer manager, and uniform distribution compared to predictions of the theoretical model.

BUFFER MANAGERS

VIRTUAL BUFFER SIZE	POI Distribution FIFO	SCH	LRU	RAND	POS Distribution FIFO	SCH	LRU	RAND
1	12500	12500	12500	12500	9236	9236	9236	9236
5	8074	7938	7775	8096	9226	9226	9231	6721
10	4372	3870	3493	4415	3725	3172	2665	3462
15	2246	1587	1382	2270	1769	1309	1153	1635
20	1097	653	554	1142	776	499	391	787

TABLE IV

The I/O accesses generated by the buffer managers in simulation experiments using the POI and POS distribution.